import java.util.Arrays;

import java.util.Scanner;

public class HeapSort {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of elements: ");

int n = scanner.nextInt();

int[] arr = ArrayInput.getUserArray(n);

System.out.println("Before sorting:");

System.out.println(Arrays.toString(arr));

heapSort(arr);

System.out.println("After sorting:");

System.out.println(Arrays.toString(arr));

scanner.close();

}

public static void heapSort(int[] arr) {

int n = arr.length;

// Build max heap

for (int i = (n / 2) - 1; i >= 0; i--) {

heapify(arr, n, i);

}

// Extract elements from the heap one by one

for (int i = n - 1; i >= 0; i--) {

// Move current root (maximum element) to the end

int temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

// Heapify the reduced heap

heapify(arr, i, 0);

}

}

public static void heapify(int[] arr, int n, int i) {

int largest = i;

int left = 2 \* i + 1;

int right = 2 \* i + 2;

// Find the largest element among the root, left child, and right child

if (left < n && arr[left] > arr[largest]) {

largest = left;

}

if (right < n && arr[right] > arr[largest]) {

largest = right;

}

// If the largest element is not the root, swap them and heapify the affected

// sub-tree

if (largest != i) {

int temp = arr[i];

arr[i] = arr[largest];

arr[largest] = temp;

heapify(arr, n, largest);

}

}

}

class ArrayInput {

public static int[] getUserArray(int n) {

Scanner scanner = new Scanner(System.in);

int[] arr = new int[n];

System.out.println("Enter the elements:");

for (int i = 0; i < n; i++) {

arr[i] = scanner.nextInt();

}

scanner.close();

return arr;

}

}